

Patent/Publication Number: JP11066041A

Application Number: JP1997231316A

Date Filed: 19970827

Title: CALCULATION PROCESSING METHOD FOR SIMULTANEOUS LINEAR EQUATIONS FOR MEMORY
DISTRIBUTED PARALLEL COMPUTER AND PARALLEL COMPUTER

Publication Date: 19990309

[INVENTOR]

Name: NAKANISHI MAKOTO

City: Country:

[ASSIGNEE]

Name: FUJITSU LTD

City: Country:

[FOREIGN PRIORITY]

Country: JP

Date Filed: 19970827

Application No.: JP1997231316A

Intl. Class: G06F001712

Intl. Class: G06F001716

[ABSTRACT]

PROBLEM TO BE SOLVED: To solve simultaneous linear equations at high speed by using a comparatively small memory while efficiently performing transfer between processors.

SOLUTION: Blocks arranged at processors are cyclically selected and numbered (1) and these numbered blocks are cyclically processed in the order of numbers so that LU decomposition is performed (2). In the case of matrix product calculation in the LU decomposition, data as the calculation object of matrix product are divided and transferred to the respective processors and by simultaneously performing the calculation of the matrix product at each processor to the divided data and parallel transfer, transfer time is shortened. Next, each processor performs forward substitution concerning the LU decomposed result and transfers the result to the adjacent processor (3). Besides, backward substitution is similarly performed as well. When transferring the result of forward/backward substitution to the other processor, the excessive data of one bit are transferred at least in addition to the transfer data and from these one-bit data, the reception of data is confirmed at the respective processors. COPYRIGHT: (C)1999, JPO&Japio

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-66041

(43) 公開日 平成11年(1999) 3月9日

(51) Int.Cl.⁶G 0 6 F 17/12
17/16

識別記号

F I

G 0 6 F 15/324
15/347

K

審査請求 未請求 請求項の数4 O L (全 11 頁)

(21) 出願番号 特願平9-231316

(22) 出願日 平成9年(1997) 8月27日

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 中西 誠

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74) 代理人 弁理士 長澤 俊一郎 (外1名)

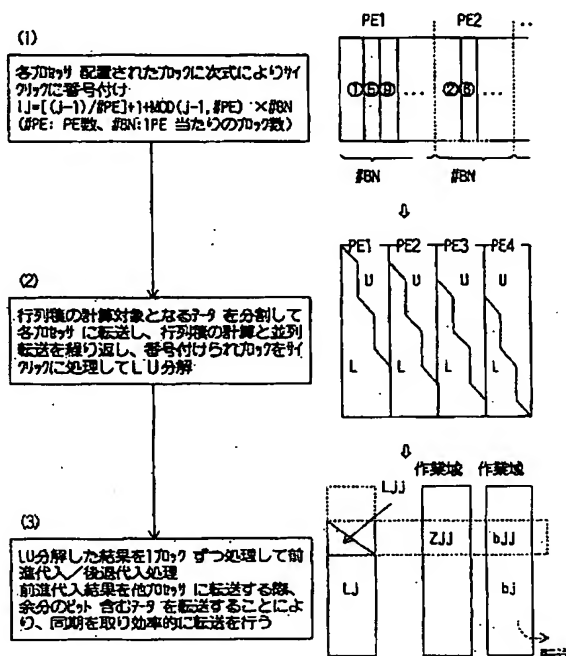
(54) 【発明の名称】 メモリ分散型並列計算機による連立一次方程式の計算処理方法および並列計算機

(57) 【要約】

【課題】 プロセッサ間で効率のよい転送を行いながら、比較的小さなメモリを使用して高速に連立一次方程式を解くこと。

【解決手段】 プロセッサに配置されたブロックをサイクリックに選択して番号付けし、上記番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解を行う。LU分解における行列積の計算の際、行列積の計算対象となるデータを分割して各プロセッサに転送し、分割したデータに対する各プロセッサにおける行列積の計算と並列転送を同時に行い転送時間を短縮する。ついで、LU分解した結果について各プロセッサで前進代入し、結果を隣接するプロセッサに転送する。また、後退代入も同様に行う。前進代入／後退代入結果を他のプロセッサに転送する際、転送データに加えて少なくとも1ビットの余分なデータを転送し、該1ビットデータにより各プロセッサにおいてデータ受信を確認する。

本発明の原理説明図



【特許請求の範囲】

【請求項1】 各プロセッサ間でデータ転送を行うことができる複数のプロセッサを備えたメモリ分散型並列計算機を用い、各プロセッサに係数行列を分配し、ブロック化した外積形式のLU分解法により連立一次方程式を解くメモリ分散型並列計算機による連立一次方程式の計算処理方法であって、
列ベクトルを束ねたブロックが各プロセッサに分散して配置されているとして、プロセッサに配置されたブロックをサイクリックに選択して番号付けし、
上記番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解を行い、
上記LU分解した結果について各プロセッサで前進代入を行って結果を隣接するプロセッサに転送するとともに、各プロセッサで後退代入を行って結果を隣接するプロセッサに転送する処理を行うことを特徴とするメモリ分散型並列計算機による連立一次方程式の計算処理方法。

【請求項2】 各プロセッサに配置されたブロックについてLU分解を行う途中で行列積の演算を行う際、行列積の計算対象となるデータを分割して各プロセッサに転送し、
分割したデータに対する各プロセッサにおける行列積の計算と、他の分割したデータについての次の行列積の計算に用いる部分の並列転送を同時に行うことにより、転送時間を短縮することを特徴とする請求項1のメモリ分散型並列計算機による連立一次方程式の計算処理方法。

【請求項3】 前進／後退代入の結果を次の前進／後退代入を行うプロセッサに転送するため、各プロセッサに、転送データ長より少なくとも1ビットだけ大きい送信バッファおよび受信バッファを設け、
前進／後退代入の結果を他のプロセッサに転送する際、上記送信バッファに転送データを格納するとともに送信バッファの上記1ビットの領域に所定の値を設定し、全プロセッサで順次上記バッファに格納されたデータの転送を行い、
受信側プロセッサにおいて、上記受信バッファの上記1ビットの領域が所定の値に設定されたことでデータの受信を確認することを特徴とする請求項1または請求項2のメモリ分散型並列計算機による連立一次方程式の計算処理方法。

【請求項4】 各プロセッサ間でデータ転送を行うことができる複数のプロセッサを備え、各プロセッサに係数行列を分配し、ブロック化した外積形式のLU分解法により連立一次方程式を解くメモリ分散型並列計算機であって、
列ベクトルを束ねたブロックが各プロセッサに分散して配置されているとして、プロセッサに配置されたブロックをサイクリックに選択して番号付けをする手段と、
LU分解における行列積の計算の際、行列積の計算対象

となるデータを分割して各プロセッサに転送し、分割したデータに対する各プロセッサにおける行列積の計算と、他の分割したデータについての次の行列積の計算に用いる部分の並列転送を同時に行う処理を繰り返すことにより、上記番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解するLU分解手段と、
上記LU分解した結果について各プロセッサで前進代入を行って結果を隣接するプロセッサに転送するとともに、各プロセッサで後退代入を行って結果を隣接するプロセッサに転送する処理を行い、前進／後退代入の結果を他のプロセッサに転送する際、転送データ長より少なくとも1ビットだけ大きい送信バッファに転送データを格納するとともに上記1ビットの領域に所定の値を設定し、全プロセッサで順次上記バッファに格納されたデータの転送を行い、受信側プロセッサにおいて、受信バッファの上記1ビットの領域が所定の値に設定されたことでデータの受信を確認する前進代入／後退代入処理手段とを備えたことを特徴とするメモリ分散型並列計算機。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、複数のプロセッサ間で通信を行って処理を進めるマルチプロセッサにより、高速に連立一次方程式を解くことのできるメモリ分散型並列計算機による連立一次方程式の計算処理方法およびメモリ分散型並列計算機に関する。

【0002】

【従来の技術】並列処理により連立一次方程式を解くアルゴリズムとして、ブロック化した外積型のLU分解法が知られている。上記方法は、外積形式のガウスの消去法をブロック化した方法であり、図16に示す配列AをLU分解する場合、ブロック幅をdとすると、以下のような処理を行う。

【0003】k番目の処理で、更新部分 $A^{(k)}$ を次の計算で更新する。

$$A^{(k)} = A^{(k)} - L_2^{(k)} \cdot U_2^{(k)} \quad (1)$$

k+1番目の処理では、 $A^{(k)}$ を幅dで分解してdだけ小さいマトリックスを同じ式で更新する。 $L_2^{(k)}$ 、 $U_2^{(k)}$ は以下の式で求める必要がある。上記(1)式で更新を行うとき、 $B^{(k)} = [(L_1^{(k)})^T, (L_2^{(k)})^T] U_1^{(k)}$ と分解し、 $U'^{(k)} = (L_1^{(k)})^{-1} \cdot U_2^{(k)}$ と更新する。

【0004】上記ブロック化した外積型のLU分解法をメモリ分散型並列計算機で実行する場合には、各プロセッサの負荷ができるだけ均等になるように各プロセッサのメモリにデータを効率よく分配し、プロセッサ間で処理対象データの交換を効率よく行う必要がある。また、問題を解くユーザインタフェースを煩雑にせずに提供する必要がある。

【0005】そこで、本発明者は、先に、データ転送コストを削減し、並列性を高めて高速化を図ることのできる

るメモリ分散型並列計算機による連立一次方程式の計算処理方法を提案した（特開平 7 - 2 7 1 7 6 0 号公報参照）。上記処理方法は、列ベクトルを束ねたブロックレベルで、データをサイクリックな配置に並列転送で並べかえ、LU分解時に、行列積の計算と、データ転送を並列的に同時に実行する。そして、LU分解した結果についてデータを元の配置に戻し、さらに行列を行ベクトル方向に分割した配置になるように並び換え、並列に前進／後退代入処理を行うものである。

【0006】

【発明が解決しようとする課題】上記従来の計算処理方法は次のような問題点を有していた。

(1) 列ブロックをサイクリックに動的に並び換えており、そのための作業域として、係数行列を格納する分だけ余分に持つ必要があった。

(2) LU分解した結果についてデータを元の配置に戻し、前進／後退代入処理を行う際、さらに行列を行ベクトル方向に分割した配置になるように並び換える必要があった。

本発明は上記した事情を考慮してなされたものであって、列ブロックを並び換えることなく、各計算機に分割されて配置されたブロックをサイクリックに処理することにより、比較的小さなメモリを使用して高速に連立一次方程式を解くことができ、また、プロセッサ間で効率のよい転送を行うことができるメモリ分散型並列計算機による連立一次方程式の計算処理方法および計算機を提供することである。

【0007】

【課題を解決するための手段】図1は本発明の原理構成図である。同図に示すように、本発明はメモリ分散型並列計算機を用い、ブロック化した外積形式のLU分解法により連立一次方程式を次のようにして解く。

(1) 列ベクトルを束ねたブロックが各プロセッサに分散して配置されているとして、プロセッサに配置されたブロックをサイクリックに選択して番号付けし、上記番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解を行う。

(2) LU分解における行列積の計算の際、行列積の計算対象となるデータを分割して各プロセッサに転送し、分割したデータに対する各プロセッサにおける行列積の計算と、他の分割したデータについての次の行列積の計算に用いる部分の並列転送を同時に行う処理を繰り返すことにより、上記番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解する

(3) 上記LU分解した結果について各プロセッサで前進代入し、結果を隣接するプロセッサに転送する。また、後退代入も各プロセッサのブロックを後退代入して、結果を隣接するプロセッサに転送する。前進代入／後退代入結果を他のプロセッサに転送する際、転送データ長より少なくとも1ビットだけ大きい送信バッファに

転送データを格納するとともに上記1ビットの領域に所定の値を設定し、全プロセッサで順次上記バッファに格納されたデータの転送を行い、受信側プロセッサにおいて、受信バッファの上記1ビットの領域が所定の値に設定されたことでデータの受信を確認することにより転送の同期をとる。

【0008】本発明においては、分割された領域に均等に格納された係数行列のLU分解を行うに際し、ピボット列を残りのどのベクトルから選んでもよいことに着目し、上記のようにプロセッサに配置されたブロックをサイクリックに選択して番号付けし、番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解を行っている。このため、従来例のように列ブロックをサイクリックに動的に並べ換えるための作業域を係数行列を格納する分だけ余分に持つ必要がない。その結果、同じメモリを使用して、約 $\sqrt{2}$ 倍の問題を解くことができる。

【0009】また、本発明においては、LU分解における行列積の計算の際、行列積の計算対象となるデータを分割して各プロセッサに転送しており、分割したデータに対する各プロセッサにおける行列積の計算と、他の分割したデータについての次の行列積の計算に用いる部分の並列転送を同時に行うことができ、見かけの転送時間は、計算と同時に進行することができない最初の転送時間のみとなる。このため、従来の2分木転送に比べ、大幅に転送時間を短縮することができる。すなわち、従来の2分木転送では $\text{LOG } 2$ (#PE) に比例した転送時間が掛かるが、本発明においては、プロセッサ数が多くなった場合、計算を行うに必要な転送時間をほぼ1（1つのプロセッサから各プロセッサにデータを転送する時間が、1つのプロセッサから他の1つのプロセッサにデータを転送するに要する時間と同程度の時間となる）とすることができる。

【0010】さらに、本発明においては、LU分解された結果が各プロセッサにサイクリックに配置されており、前記した従来例のようにLU分解された結果を行ベクトル方向に並べ換えることなく、i番目のブロックがあるプロセッサで前進代入を行って結果を次の前進代入を行うプロセッサに転送し、後退代入についても、同様に各プロセッサのブロックを後退代入して、結果をプロセッサに転送することにより先進代入／後退代入処理を行う。また、上記データ転送を行う際、前記したように転送データに加えて少なくとも1ビットの余分なデータを転送し、該1ビットデータによりデータ受信を確認しているので、効率的な転送を行うことができる。

【0011】

【発明の実施の形態】図2は本発明に用いられるメモリ分散型並列計算機の構成の一例を示す図であり、図3は図2における各プロセッサ（以下PEという）の構成の一例を示す図である。図2において、各プロセッサ（P

E) 1はクロスバーネットワーク6に接続されている。各プロセッサ1は、それぞれスカラ演算を行うスカラユニット2と、ベクトル演算を行うベクトルユニット3と、主記憶装置4と、PE間通信ユニット5から構成され、PE間通信ユニット5により上記クロスバーネットワーク6を介して任意の他のプロセッサとの間で通信を行う。

【0012】各プロセッサ1は図3に示すように構成されている。各スカラユニット2は主記憶装置4のデータを一時的に保持するキャッシュメモリ11と、演算に用いる汎用レジスタ/浮動小数点レジスタ12と、スカラ命令を実行するスカラ演算機13から構成される。また、ベクトルユニット3は、主記憶装置4からデータをロードするためのロードパイプライン14と、主記憶装置4へデータをストアするためのストアパイプライン15と、ベクトル演算の対象となる一連のデータを保持するベクトルレジスタ16と、特定の演算対象をマスクするマスクレジスタ17と、演算対象データを指定するマスクパイプライン18と、ベクトルデータの乗算を実行する乗算パイプライン19と、ベクトルデータの加減算または論理演算を実行する加算/論理演算パイプライン20と、ベクトルデータの除算を行う除算パイプライン21から構成される。そして、主記憶装置4がフェッチした命令がベクトル命令であるときに上記ベクトルユニット3が起動され、ベクトル演算が実行される。

【0013】次に、本発明の実施例の連立一次方程式の解法について説明する。

(1) LU分解

行列Aに対する連立一次方程式 $Ax=b$ を解くことを考える。ここで、Aの列ベクトルをd本束ねたものをブロックと見なし、この列ベクトルを束ねたブロックを下記のように A_1, A_2, \dots, A_m とする。

$$A = [A_1, A_2, \dots, A_m]$$

A_j を並べ換えて生成した行列Eを下記のように表す。

$$F(j) = [(L_1(j))^T, (L_2(j))^T] U_1(j) \dots (2)$$

と分解し、

$$U'_{2(K)} = (L_1(j))^{-1} \cdot U_2(j) \dots (3)$$

と更新する。

【0017】そして、 $E(j)$ を次のように更新する。

$$E'(j) = E(j) - L_2(j) \cdot U'_{2(j)} \dots (4)$$

ここで、 $L_1(j)$ 、 $U_1(j)$ 、 $L_2(j)$ は同一プロセッサ上にあるが、 $U_2(j)$ および $E(j)$ は各プロセッサに分散されており、各々のプロセッサ上で計算する必要がある。そこで、 $L_1(j)$ 、 $L_2(j)$ を全プロセッサに通信して前記(1)式により $ij > j$ なるブロックに関して(3)(4)式の計算を行う。

【0018】図6は上記処理を行うフローチャートである。同図において、ステップS1において、 $j=1$ に設定し、ステップS2において、行列Aの前記(1)式で求めた ij 番目のブロックを前記(2)式のように分解

$$E = [A_{i1}, A_{i2}, \dots, A_{im}]$$

つまり、 $AP=E$ であり、 $Ax=b$ は下記のように表すことかできる。但し、Pは列ブロックの並べ換えを表す。

$$APP^{-1}x=b$$

$$EP^{-1}x=b$$

【0014】ここで、 $P^{-1}x=y$ と置くと、 $Ax=b$ は下記の式を解くことに相当する。

$$Ey=b, P^{-1}x=y$$

行列の次数をnとし、各プロセッサに等しい数のブロックが配置されるとすると、nは次のように表される。

$$n = \#BN \times \#PE \times d$$

但し、 $\#PE$ は並列プロセッサ数、 $\#BN$ は各プロセッサに配置される行列のブロック数、dは各ブロックに含まれる列ベクトルの数である。

【0015】いま、 $E = [A_{i1}, A_{i2}, \dots, A_{im}]$ の i_j ($j=1, \dots, m$)が次の式で与えられる場合について考える。

$$i_j = [(j-1)/\#PE] + 1$$

$$+ \text{MOD}(j-1, \#PE) \times \#BN \dots (1)$$

ここで、 $\text{MOD}(a, b)$ は整数aを整数bで割ったときの剰余を表す。この並び換えに対する P^{-1} は、ブロックをサイクリックに番号付けすることに対応する。すなわち、前記(1)式で i_j 番目の行列Aのブロックに番号jを振ることに対応し、各プロセッサのブロック $\#BN$ は図4に示すように番号付けされることになる。

【0016】ここでは、以上のように番号付けたブロックを並べ換えて作った行列EをLU分解することを考えるが、実際には、行列Aを上記(1)式で決まる番号のブロックに着目してLU分解を行うことになる。行列Eのj番目のブロックに関する分解は前記したのと同様、図5に示すようになる。すなわち、行列Eのj番目のブロックに関する分解は、

する。なお、このブロックは、 $\text{MOD}(j-1, \#PE) + 1$ 番目のプロセッサ(PE)上にある。ステップS3において、次のステップS4の計算をするため、 $L_1(j)$ を全プロセッサに通信する。ついで、ステップS4において、各プロセッサにおいて、前記(3)式により $U'_{2(j)}$ を更新する。ステップS5において、 $L_2(j)$ を全プロセッサに通信し、ステップS6において、 $ij > j$ なるブロックに対して、前記(4)式により、各プロセッサに割り付けられた部分を計算する。ステップS7において $j=j+1$ とし、ステップS8において、 $j > \#BN \times \#PE$ であるかを調べ、 $j > \#BN \times \#PE$ でない場合には、ステップS2に戻り上記処理を繰り返す。また、 $j > \#BN \times \#PE$ の場合には処理を終了する。

【0019】その結果、各プロセッサに配置されたプロ

ックは、図7に示すようにLU分解される。同図において、斜線部分がL、斜線の無い部分がUである。以上のような処理を行うことにより、前記したように、データの並べ換えを行うことなくLU分解を行うことができ、並べ換えの作業域として、係数行列を格納する分だけ余分に持つ必要がなくなり、比較的小さなメモリを使用して連立一次方程式を解くことができる。

【0020】ところで、上記処理においてはデータを各プロセッサ（PE）に転送して行列積の計算を行っているが、その際、転送時間を短縮し転送と計算を同時に行うため、前記した特開平7-271760号で示したように、各プロセッサ（PE）にデータを分割して転送する。そして、分割したデータに対する計算を各プロセッサ（PE）で行い、これを繰り返して全体の計算を行う。これにより、実際の転送時間が非常に少なくなったように見える。

【0021】以下、簡単に上記転送・計算処理について説明する。なお、詳細は上記特開平7-271760号公報を参照されたい。前記したように、ブロック化したLU分解のj番目のステージにおいて、前記した（4）式に示した行列積の演算を行いE(j)を更新している。ここでは、プロセッサ数を5として、 $U = U - C \times R$ の計算を行ってUを更新する場合を考える。図8に示すように、行列Cを行方向に分割し、例えばC1、C2、C3とする。なお、上記分割数nは、 $\#PE / n > 1$ で $LOG_2(\#PE / n) < n$ になるように定める。ここでは分割数を3とする。また、各プロセッサに第1のワーク領域W11～W51、第2のワーク領域W12～W52を設ける。

【0022】図8において、PE1におけるCの部分の計算が完了したら、C1をPE1のW11へ、C2をPE2のW21へ、C3をPE3のW31へそれぞれ転送する。次に、その結果を使ってW11のデータ（C1）をPE4のW41へ、W21のデータ（C2）をPE5のW51へ並列転送する。行列式の計算は、最初に各PEの第1のワーク領域W11～W51に格納されたCiを用いて計算を行う。図8においては、ハッチングで示した部分が最初に計算する部分である。すなわち、PE1において $C1 \times R1$ の行列積が計算され、PE2において $C2 \times R2$ の行列積が計算され、PE3において $C3 \times R3$ の行列積が計算され、PE4において $C1 \times R4$ の行列積が計算され、さらに、PE5において $C2 \times R5$ の行列積が計算される。

【0023】上記計算と同時に、図8に示すようにワーク領域W11からW22へ、W21からW32へ、W31からW42へ、W41からW52へ並列にデータ転送を行うとともに、PE1が保持するC3をW12に転送する。次に、各PEの第2のワーク領域W12～W52に格納されたCiを使って、図9に示すハッチングの部分を計算する。すなわち、PE1において $C3 \times R1$ の

行列積が計算され、PE2において $C1 \times R2$ の行列積が計算され、PE3において $C2 \times R3$ の行列積が計算され、PE4において $C3 \times R4$ の行列積が計算され、さらに、PE5において $C1 \times R5$ の行列積が計算される。

【0024】上記計算と同時に、図9に示すようにワーク領域W12からW21へ、W22からW31へ、W32からW41へ、W42からW51へ並列にデータ転送を行うとともに、PE1が保持するC2をW11に転送する。さらに、各PEの第1のワーク領域W11～W51に格納されたCiを使って、図10に示すハッチングの部分を計算する。すなわち、PE1において $C2 \times R1$ の行列積が計算され、PE2において $C3 \times R2$ の行列積が計算され、PE3において $C1 \times R3$ の行列積が計算され、PE4において $C2 \times R4$ の行列積が計算され、さらに、PE5において $C3 \times R5$ の行列積が計算される。

【0025】以上のようにしてPE1に保持されたC1～C3についての行列積の計算を終了すると、ついで、図11のハッチングで示す部分について、LU分解して、上記と同様、行列積の計算を行う。すなわち、図12に示すようにPE2に保持されたC1～C3を3分割し、C1をPE2のワーク領域W21へ、C2をPE3のW31へ、C3をPE4のW41へそれぞれ転送し、その結果を使って、W21（C1）のデータをPE5のW51へ、W31（C2）のデータをPE1のW11に転送する。

【0026】次に、前記したようにして各PEの第1のワーク領域W11～W51に格納されたCiを使って、図12のハッチングで示した部分の計算を行い、これらの計算と同時に、図12に示すように、ワーク領域W21からW32へ、W31からW42へ、W41からW52へ、W51からW12へ並列にデータ転送を行うとともに、PE2が保持するC3をワーク領域W22に転送する。以下、同様に計算と転送を繰り返し、LU分解を完了する。

【0027】以上のように、行列積部分の計算に必要なデータを分割して、転送・計算することにより、転送の大部分を計算と同時に行うことができる。この場合、計算と転送を同時に行うことができない最初の転送のみが見掛け上の転送時間となるが、この転送も並列に行うようにしたため、従来の2分木転送に比べ大幅に転送時間を短縮することができる。その結果、単純に行列積の計算で必要なデータを各プロセッサに2分木のパターンで転送する場合の転送時間が $LOG_2(\#PE)$ に比例するのに比べ、本実施例では、 $1 + [LOG_2(\#PE / \#div)] / \#div$ （ $\#div$ は分割数）のオーダーになる。上記式の第2項は0.5以下にすることができるので、プロセッサ数が2台以上あるシステムで、特にプロセッサ数が大きくなった場合に効率を著しく向上さ

ることができる。

【0028】(2) 前進代入／後退代入処理

上記LU分解の処理結果は、図6に示したように行列Aのブロックに前記(1)式で指定される順番に格納されている。本実施例の前進代入／後退代入処理では、前記した特開平7-271760号公報のように行列を行ベクトル方向に並び換えずに、1ブロックずつ処理をする。すなわち、LU分解した $LUy=b$ を、 $Lz=b$ 、 $Uy=z$ とし、この順に解く。 $Lz=b$ を解くとき、 j が1から $\#BN \times \#PE$ まで順に動き、次のような計算を行う。各プロセッサに、図13に示すように上記 z および b を格納するためのワーク領域 Wz 、 Wb を設け、このワーク領域を利用して計算を行う。

【0029】まず、 $L_{jj} \cdot Z_{jj} = b_{jj}$ を解き、次に $b_j = b_j - L_j \times Z_{jj}$ により b_j を更新する。すなわち、 $PE1$ で $L_{11} \cdot Z_{11} = b_{11}$ を解き、 $b_1 = b_1 - L_1 \times Z_{11}$ により b_1 を更新して、次いで、 $PE2$ において、 $j=2$ について同様の処理を行い、以下同様に各プロセッサにおいて上記処理を繰り返す。上記 b_j は $j+1$ 番目の処理で必要となるため、他のプロセッサに転送する必要がある。

【0030】この転送は、 $PE1$ から $PE2$ へ、 $PE2$ から $PE3$ へ、…、 $PE\#PE$ ($\#PE$ はプロセッサ数)から $PE1$ へと循環するパターンで行われる。上記転送を行うため、各プロセッサに、上記 b と同じ大きさのバッファ bT を設け、 j 番目の前進代入において、バッファ bT の $1+(j-1)/\#PE \times d$ 番目のデータから最後のデータまで($b(1+(j-1)/\#PE \times d:n)$)を循環的に同時に転送する。その際、以下のようにして転送の同期を取る。

【0031】転送するデータ長は全プロセッサで一定であり、その長さを L とすると、図14に示すように、各プロセッサに $L+1$ の受信バッファ、送信バッファを設け、全プロセッサで $L+1$ のデータを転送する。その際、送信バッファの $L+1$ 番目には1を設定しておき、受信バッファの $L+1$ 番目には0を設定しておく。そして、転送時、全プロセッサで次の図15に示す処理を行い、データを受け取ったかを確認する。図15において、ステップS1においてバリア同期を取り、ステップS2において、転送データ長 L を決めて、図14(a)に示すように受信バッファの $L+1$ 番目を0クリアし、送信バッファの $L+1$ 番目に1を設定する。ステップS3において、全プロセッサで $L+1$ のデータを送り、ステップS4において、受信バッファの $L+1$ 番目が1であるかを調べ、図14(b)に示すように受信バッファの $L+1$ 番目が1になるとデータを受け取ったとして、次の処理を行う。

【0032】上記のようにして前進代入を行った後、後退代入においても、各プロセッサにブロックを後退代入して、結果を隣接するプロセッサに転送することで解

く。以上のように、本実施例においては前進代入／後退代入処理において、長さ L のデータの転送する際、全プロセッサで $L+1$ 番目の余分のデータを転送し、受信したプロセッサにおいて、 $L+1$ 番目のデータを参照してデータ受信を確認しているので、効率のよい転送を行うことができる。

【0033】

【発明の効果】以上説明したように、本発明においては以下の効果を得ることができる。

(1) プロセッサに配置されたブロックをサイクリックに選択して番号付けし、番号付けられたブロックを、番号順にサイクリックに処理をしてLU分解を行うことによりプロセッサの負荷を均等化している。このため、列ブロックをサイクリックに動的に並べ換えるための作業域を係数行列を格納する分だけ余分に持つ必要がなく、同じメモリを使用して、約 $\sqrt{2}$ 倍の問題を解くことができる。

【0034】(2) LU分解における行列積の計算の際、行列積の計算対象となるデータを分割して、転送・計算を行っており、転送の大部分を計算と同時に行うことができ、見かけの転送時間は、計算と同時に行うことができない最初の転送時間のみとなる。このため、従来の2分木転送に比べ、大幅に転送時間を短縮することができ、計算を行うに必要な転送時間をほぼ1(1つのプロセッサから各プロセッサにデータを転送する時間が、1つのプロセッサから他の1つのプロセッサにデータを転送するに要する時間と同程度の時間となる)とすることができる。特にプロセッサ数が多くなった場合に非常に効率よい処理を行うことができる。

(3) 先進代入／後退代入処理において、データを他のプロセッサに転送する際、転送データに加えて少なくとも1ビットの余分なデータを転送し、該1ビットデータによりデータ受信を確認しているので、効率的な転送を行うことができる。

【図面の簡単な説明】

【図1】本発明の原理説明図である。

【図2】本発明の実施例のメモリ分散型並列計算機の構成例を示す図である。

【図3】図2における各プロセッサの構成を示す図である。

【図4】各プロセッサに配置されたブロックへの番号付けを説明する図である。

【図5】本発明の実施例におけるLU分解を説明する図である。

【図6】本発明の実施例におけるLU分解処理のフローチャートである。

【図7】本発明の実施例による行列AのLU分解結果を示す図である。

【図8】本発明の実施例における行列積の計算を説明する図(1)である。

【図 9】本発明の実施例における行列積の計算を説明する図 (2) である。

【図 10】本発明の実施例における行列積の計算を説明する図 (3) である。

【図 11】本発明の実施例における行列積の計算を説明する図 (4) である。

【図 12】本発明の実施例における行列積の計算を説明する図 (5) である。

【図 13】本発明の実施例における前進代入処理を説明する図である。

【図 14】前進代入／後退代入処理時のデータ転送を説明する図である。

【図 15】前進代入／後退代入処理時のデータ転送処理を示すフローチャートである。

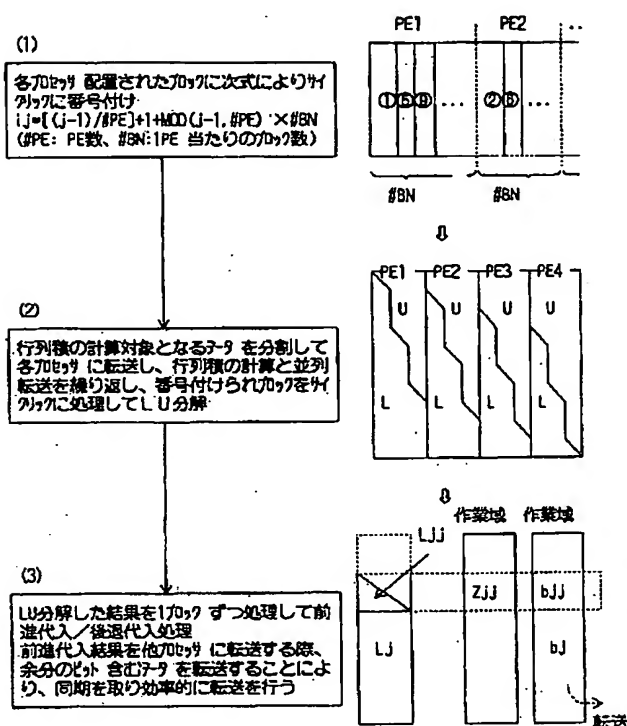
【図 16】ブロック化した外積型の LU 分解法の説明図である。

【符号の説明】

- 1 プロセッサ (PE)
- 2 スカラユニット
- 3 ベクトルユニット
- 4 主記憶装置
- 5 PE 間通信ユニット
- 6 クロスバーネットワーク
- 11 キャッシュメモリ
- 12 汎用レジスタ／浮動小数点レジスタ
- 13 スカラ演算機
- 14 ロードパイプライン
- 15 ストアパイプライン
- 16 ベクトルレジスタ
- 17 マスクレジスタ
- 18 マスクパイプライン
- 19 乗算パイプライン
- 20 加算／論理演算パイプライン
- 21 除算パイプライン

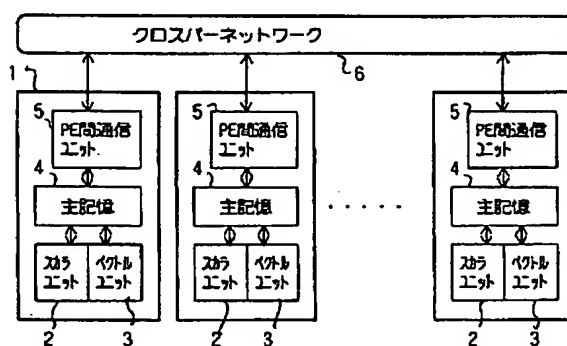
【図 1】

本発明の原理説明図



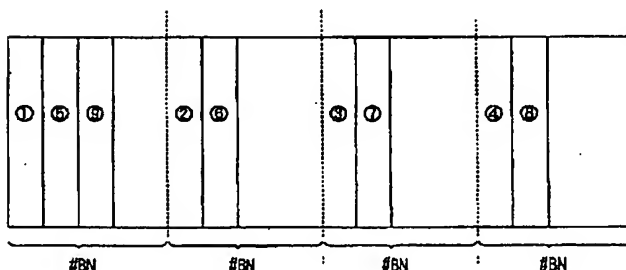
【図 2】

本発明の実施例のメモリ分散型並列計算機の構成例を示す図



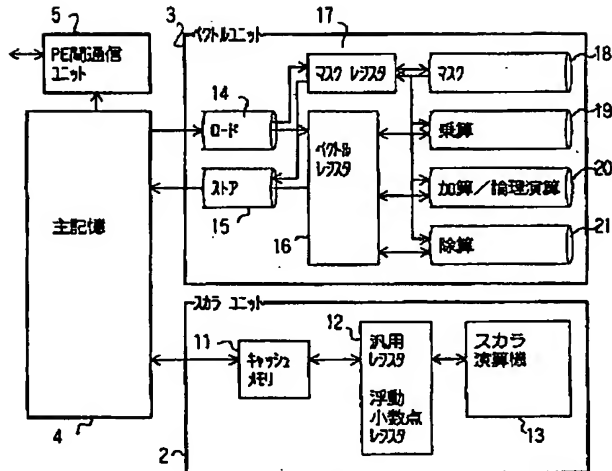
【図 4】

各プロセッサに配置されたブロックへの番号付けを説明する図



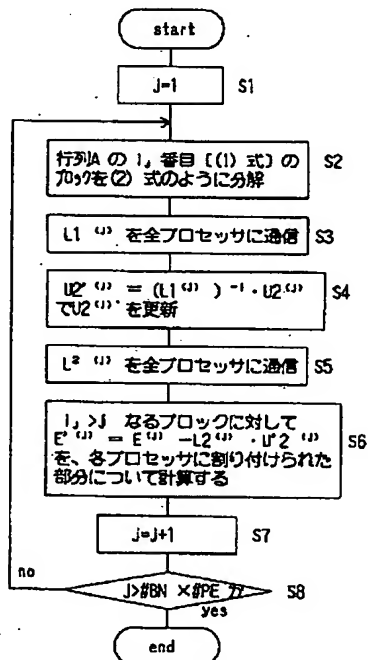
【図3】

図2における各プロセッサの構成を示す図



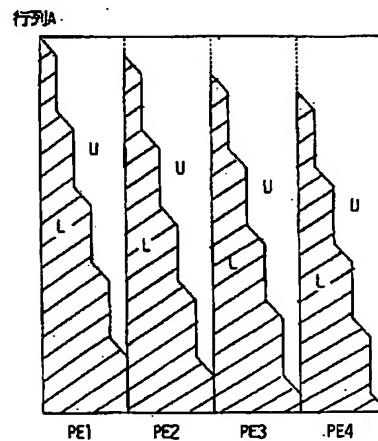
【図6】

本発明の実施例におけるLU分解処理のフローチャート



【図7】

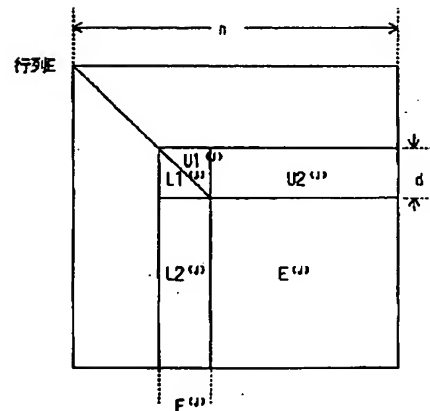
本発明の実施例による行列AのLU分解結果を示す図



【図5】

本発明の実施例におけるLU分解を説明する図

$$I_j = [(j-1)/\#PE] + 1 + \text{MOD}(j-1, \#PE) \times \#BN \quad \dots (1)$$



$$F^{(w)} = (L1^{(w)*}, L2^{(w)*})^T U1^{(w)} \quad \dots (2)$$

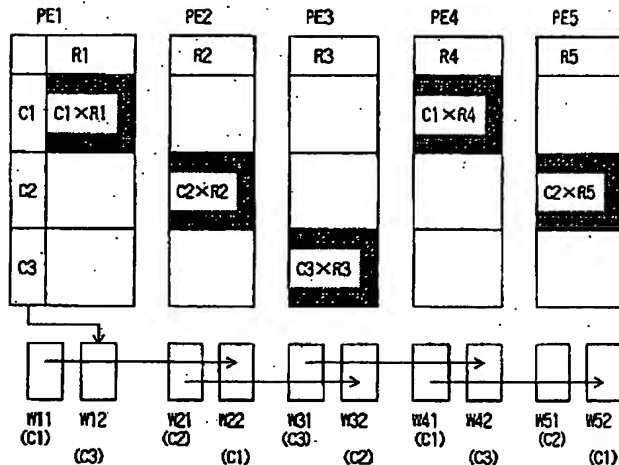
と分解して

$$U2^{(w)} = (L1^{(w)})^{-1} \cdot U2^{(w)} \quad \dots (3)$$

と更新

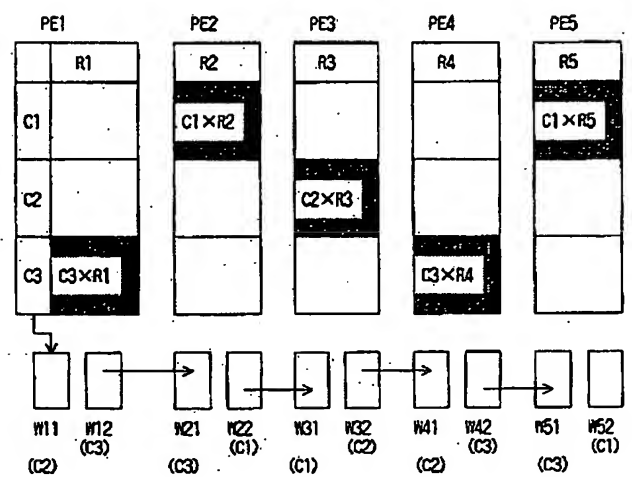
【図 8】

本発明の実施例における行列積の計算を説明する図 (1)



【図 9】

本発明の実施例における行列積の計算を説明する図 (2)

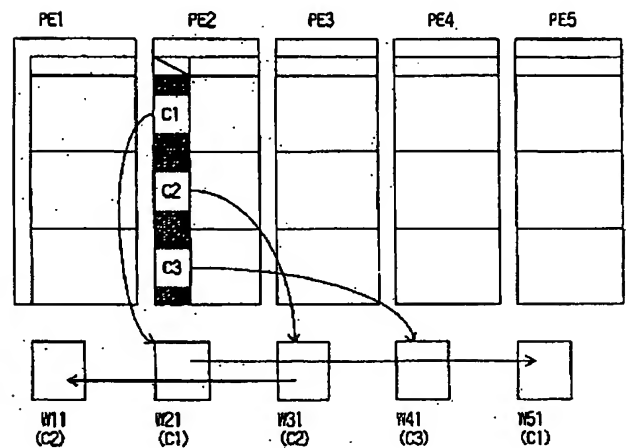
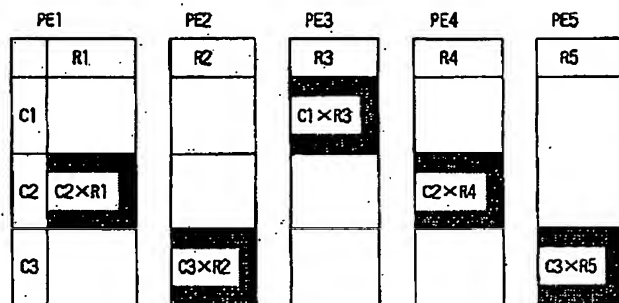


【図 1 1】

【図 1 0】

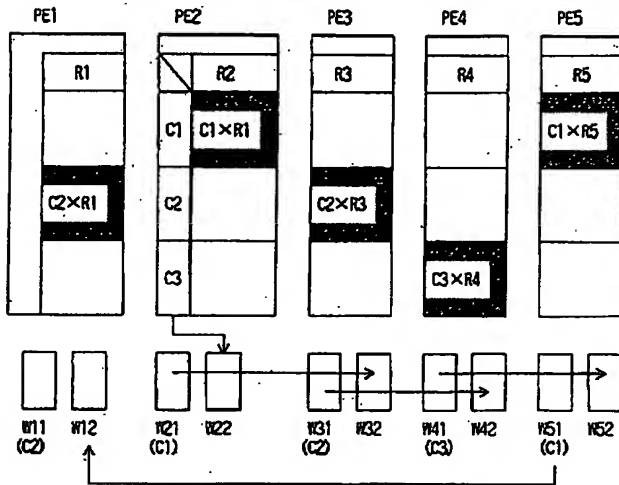
本発明の実施例における行列積の計算を説明する図 (4)

本発明の実施例における行列積の計算を説明する図 (3)



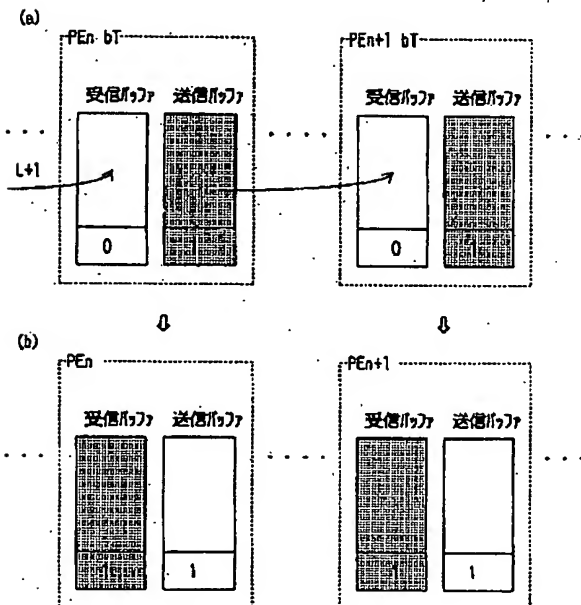
【図 12】

本発明の実施例における行列積の計算を説明する図 (5)



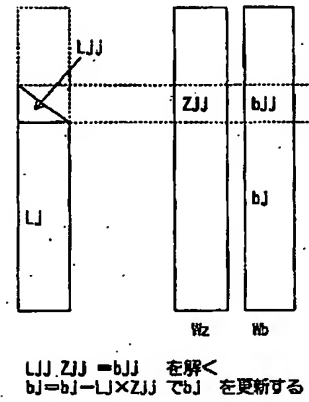
【図 14】

前進代入／後退代入処理時のデータ転送を説明する図



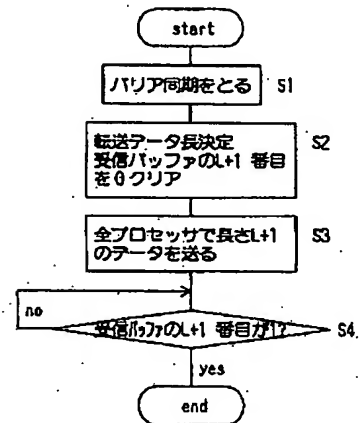
【図 13】

本発明の実施例における前進代入処理を説明する図



【図 15】

前進代入／後退代入処理時のデータ転送処理を示すフローチャート



【図 1 6】

ブロック化した外積型のLU分解法の説明図

